

# Customer training workshop: Smart I/O configurator

TRAVEO™ T2G CYT4BF series Microcontroller Training  
V1.0.0 2022-12



Please read the [Important notice and warnings](#) at the end of this document

## Scope of work

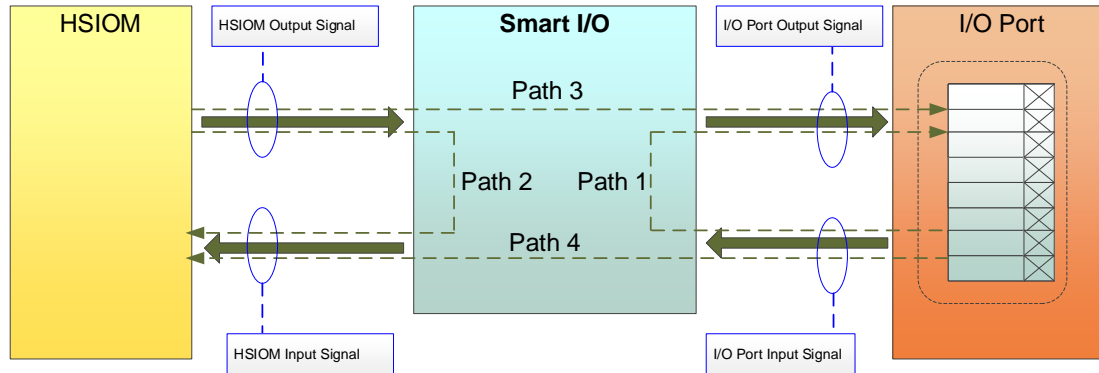
---

- › This document helps application developers understand how to use the Smart I/O Configurator as part of creating a ModusToolbox™ (MTB) application
  - The Smart I/O Configurator is part of a collection of tools included with the MTB software. It provides a GUI to configure the Smart I/O.
  
- › MTB version: 3.0.0
- › Smart I/O Configurator version: 4.0
  
- › Device
  - The TRAVEO™ T2G CYT4BFBCH device is used in this code example.
  
- › Board
  - The TRAVEO™ T2G KIT\_T2G-B-H\_EVK board is used for testing.

# Introduction

## › Smart I/O has the following features:

- Smart I/O provides the ability to perform Boolean functions in the I/O signal path
- Path1: Implement self-contained logic functions that directly operate on port I/O signals
- Path2: Implement self-contained logic functions that operate on HSIOM signals
- Path3: Operate on and modify HSIOM output signals and route the modified signals to port I/O signals
- Path4: Operate on and modify port I/O signals and route the modified signals to HSIOM input signals
- Smart I/O can be useful when the application involves simple logic operations and routing of the signal coming from or going to the I/O pin. No CPU is required for these operations.



## Introduction (contd.)

---

### › **Smart I/O has the following features:**

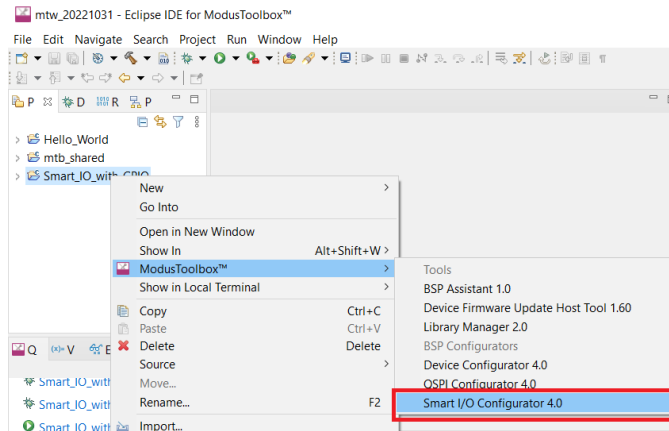
- There are several areas in the GUI to configure signals: Chip, I/O, Data Unit (DU), and LUT
- Inputs to the chip from the I/O port can be logically operated upon before being routed to the peripheral blocks and connectivity of the chip. Likewise, outputs from the peripheral blocks and internal connectivity of the chip can be logically operated upon before being routed to the I/O port.
- The programmable logic fabric of the Smart I/O can be purely combinatorial or registered with a choice of clock selection
- Each path can be selectively bypassed if certain routes are not required by the fabric
- Each Smart I/O is associated with a particular I/O port and consumes the port entirely
- If the Smart I/O is not enabled, then the Smart I/O functionality for that port is bypassed, which means that each chip terminal is routed directly to the corresponding I/O terminal.
- See the Smart I/O chapter in the Architecture technical reference manual for Smart I/O details

# Launch the Smart I/O configurator

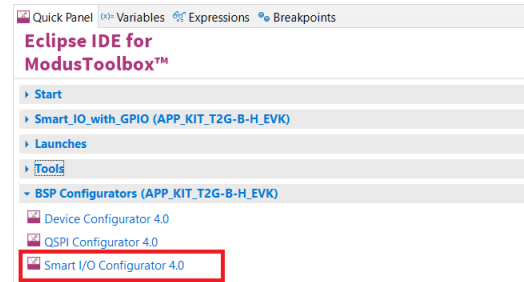
## > From Eclipse IDE

– You can launch the Smart I/O configurator by following either of these methods:

a) Right-click on the project in the Project Explorer and select ModusToolbox™ > Smart I/O Configurator <version>



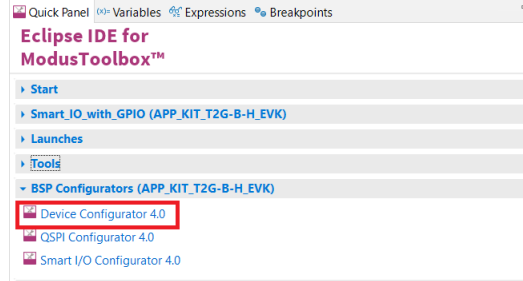
b) Click the *Smart I/O Configurator* link in the Quick Panel



# Launch the Smart I/O configurator (contd.)

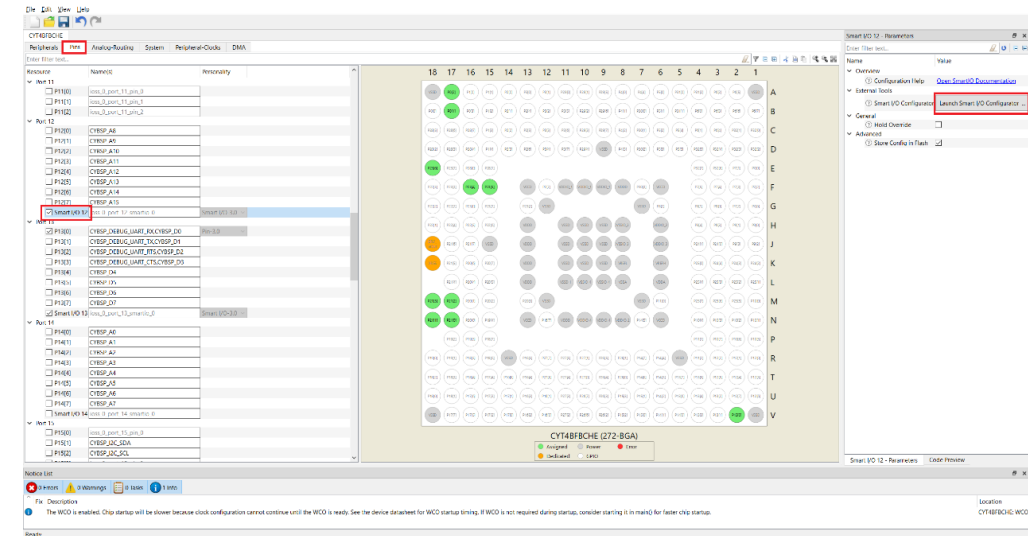
## > From Device Configurator

1) Open the Device configurator



2) On the Pins tab, enable the Smart I/O resource

3) On the Parameters tab, click the **Launch Smart I/O** Configurator button



# Smart I/O configurator

## > Smart I/O configurator view – Routing tab

**Routing** Data Unit LUT 4

Port: Port 12 (Smart I/O 12) Clock: Asynchronous Show Routing Matrix X Clear

**Clock:**  
Selects the clock source used to drive all sequential logic in the block.

**Port:**  
This parameter allows selecting a port that supports Smart I/O.

**Chip configuration:**  
Each has two pull-down menus; one to select the direction of the signal and the other to select the connection.

**Data Unit input configuration:**  
The Data Unit configuration section contains three pull-down menus to select inputs. You can click the link icon to access the tab.

**I/O configuration:**  
Each has a pull-down menu to select the direction of the signal.

**LUT input configuration:**  
Each LUT configuration section contains three pull-down menus to select inputs from Chip, I/O, and DUT resources. You can click the link icon to access the tab.

When the Show routing matrix button is on, you may click on the switches in the fabric to make input connections to the LUTs.

Ready

LUT	0	1	2
LUT 0	None	None	None
LUT 1	None	None	None
LUT 2	None	None	None
LUT 3	None	None	None
LUT 4	I/O 4	Chip 7	Chip 5
LUT 5	None	None	None
LUT 6	None	None	None
LUT 7	None	None	None

# Smart I/O configurator (contd.)

## > Smart I/O configurator view – Data Unit tab

- When the DU in the Routing tab is configured to accept an input other than a Constant 0, the corresponding Data Unit configuration tab will appear.

The screenshot shows the Smart I/O configurator interface for LUT 4. The 'Data Unit' tab is active. The configuration fields are:

- Opcode:** Increment
- DATA0:** Constant 0
- DATA1:** Constant 0
- Size:** 8-bits size/width operand

The **Details** window shows the following Verilog code:

```

Clock = Asynchronous
TR0 = LUT4
TR1 = LUT4

du_size = Size - 1
mask = (1 << (du_size + 1)) - 1
data_eq_data = (data & mask) == (DATA1 & mask)

Combinatorial:
tr_out = data_eq_data1

Registered/Clocked:
data <= data
if (TR0)
{
  data <= DATA0 & mask
}
else if (TR1)
{
  data <= data_eq_data1 ? data : (data + 1) & mask
}
  
```

The block diagram shows an 'Increment' block with inputs TR0 (rst), TR1 (en), and Clock (clk), and an output TR\_out.

### Opcode:

Defines the Data Unit operation. Each opcode performs a unique function that can be controlled using the DU trigger inputs TR0, TR1, and TR2.

Refer to the pseudo verilog code in the **Details** window and also to the Functional Description section for more information.

### DATA0:

Defines the DU DATA0 register source. This value is often used as the initial/reset value that is loaded into the DU working register when TR0 signal is high.

### DATA1:

Defines the DU DATA1 register source. This value is often used as the comparison value that gets applied to the DU working register.

### Size:

Defines the bit size operation to be performed by the data unit. Valid range is from 1 to 8 bits.

### Details:

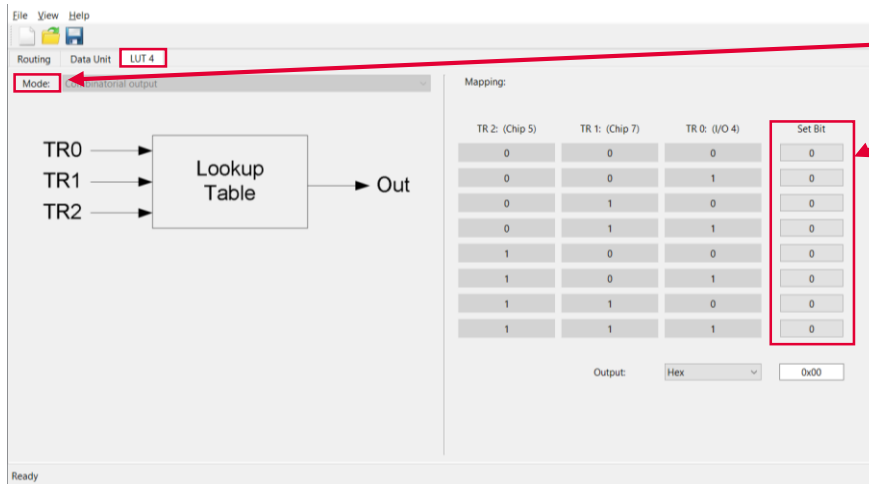
Pseudo Verilog code is shown for selected opcodes.



# Smart I/O configurator (contd.)

## > Smart I/O configurator view – LUT tab

- When a LUT in the Routing tab is configured to accept an input, the corresponding LUT configuration tab will appear.



**Mode:**  
Defines the LUT mode.

**Set Bit:**  
Defines the lookup truth table of the 3-to-1 LUT. The state on the three inputs (input 0, 1, and 2) are translated to an output value according to this truth table. Click to select 0 or 1.

If a LUT is used, all three inputs to the LUT must be designated. For example, even if a LUT is used to accept a single source as its input, all three inputs must accept that same signal. The lookup truth table should then be designed such that it only changes the output value when all three inputs satisfy the same condition (for example, it outputs a logic 1 only when the inputs are all 0).

## Quick start

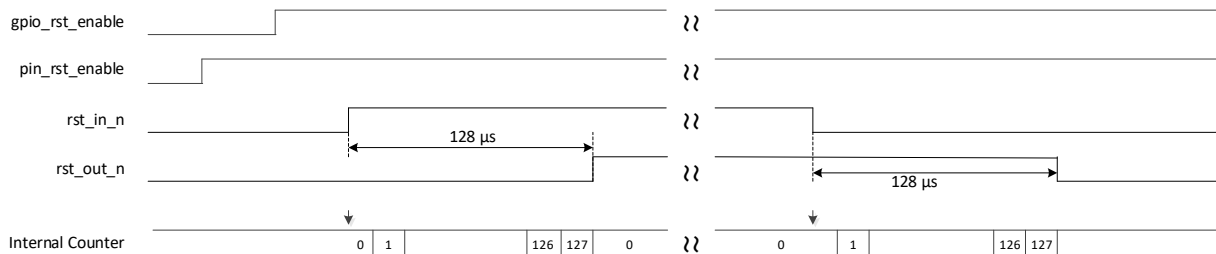
---

### › **To use the Smart I/O configurator for Smart I/O setting**

- Launch the Smart I/O Configurator.
- Use the various pull-down menus to configure signals. Refer to the descriptions in the Routing tab section for more details.
- Save the file to generate source code.
- The Smart I/O Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location you saved the *\*.modus* file for non-IDE applications. That directory contains the necessary source (.c) and header (.h) files for the generated firmware, which uses the relevant driver APIs to configure the hardware.
- Use the generated structures as input parameters for Smart I/O functions in your application.

## Use case

- › Implement a reset detection/stability circuitry on the Smart I/O
- › In this use case, the circuitry has two enable signals: `pin_rst_enable` and `gpio_rst_enable`.
- › The `pin_rst_enable` is an enable signal from an external circuitry and the `gpio_rst_enable` is an enable control signal by software. When both signals are enabled, the circuitry is active.
- › The `rst_in_n` is an external reset input with Active high and `rst_out_n` is a reset output with Active high. The circuitry monitors `rst_in_n`. When `rst_in_n` is activated for a specific number of continuous cycles (the operation clock is input continuously for 128 cycles), the `rst_out_n` is output.
- › The source 50 MHz clock is divided by 50 to 1 MHz. Then, count 1  $\mu$ s is multiplied by 128 cycles and the time of reset activation or release is approximately 128  $\mu$ s.



- › Refer to the application note [Smart I/O Usage Setup in Traveo II Family](#) for details

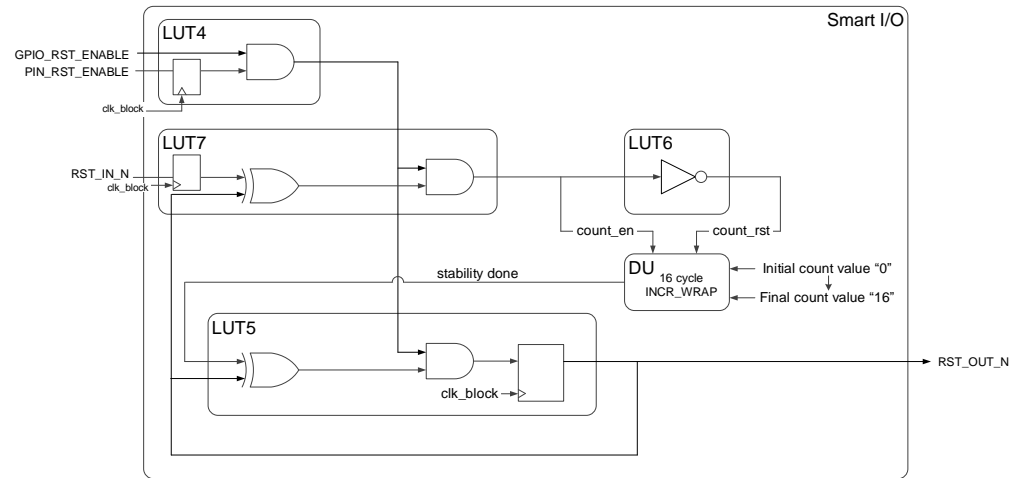
## Use case (contd.)

### > Connection and functional logic of each LUT [7:4] and DU in this circuitry

- In this example, four LUTs and one DU are used
  - LUT4 is used to generate the activation signal of this circuitry from two enable signals (pin\_rst\_enable and gpio\_rst\_en)
  - LUT6 and LUT7 are used to monitor the rst\_in\_n state and to start the counter of the DU
  - LUT5 detects the stabilization wait completion and outputs rst\_out\_n
  - DU is used to generate reset stability wait time, and the Tr\_out of LUT5 is output synchronously by the gated output mode

### - In this use case, I/O [7:5], Chip 4 is used as input or output signals

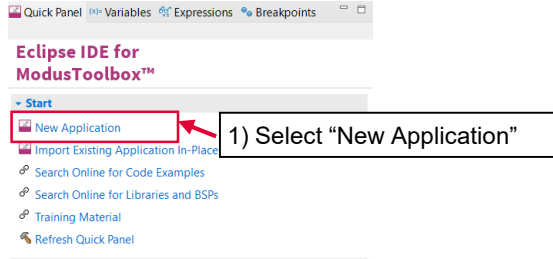
- I/O 7 is used for "RST\_IN\_N"
- I/O 6 is used for "PIN\_RST\_ENABLE"
- I/O 5 is used for "RST\_OUT\_N"
- Chip 4 is used for "GPIO\_RST\_ENABLE"



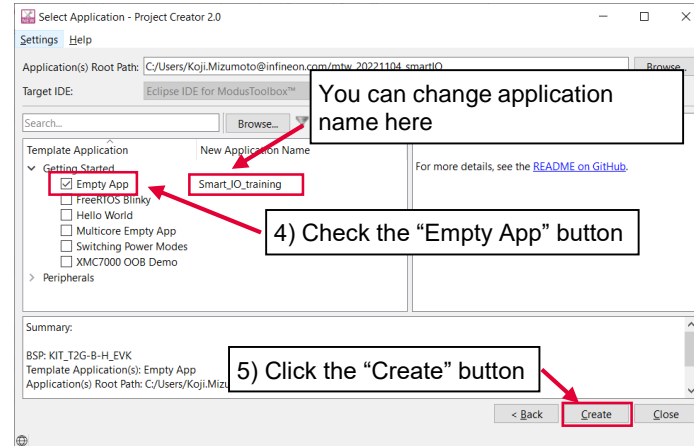
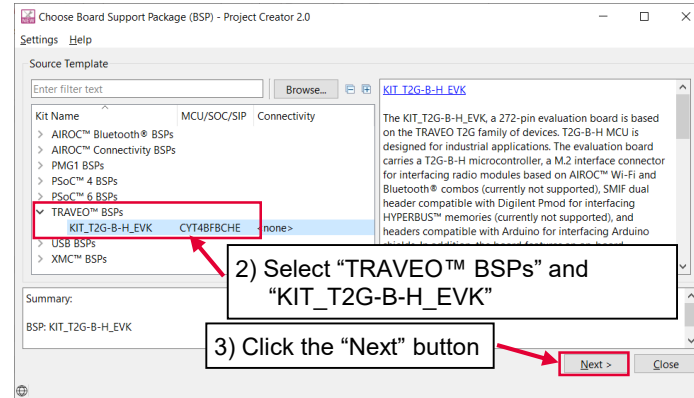
# Smart I/O configuration

## > Create project

- 1) Click “New Application” in Quick Panel and open the Choose Board Support Package (BSP) window



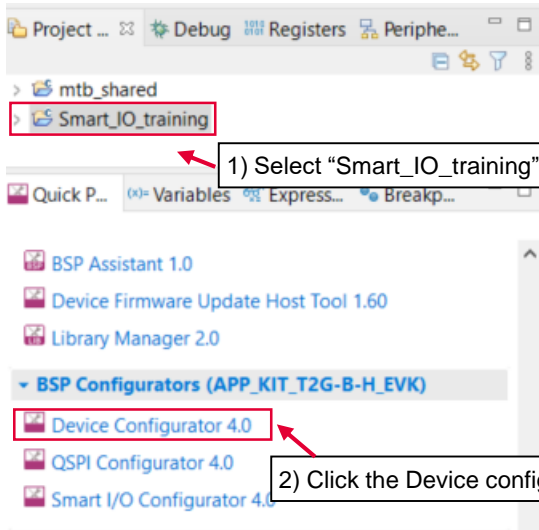
- 2) Select “TRAVEO™ BSPs” and “KIT\_T2G-B-H\_EVK”
- 3) Click the **Next** button and open the Application window
- 4) Check the “Empty App” option.  
In this use case, change the application name to “Smart\_IO\_training”.
- 5) Click the **Create** button to start application creation



# Smart I/O configuration (contd.)

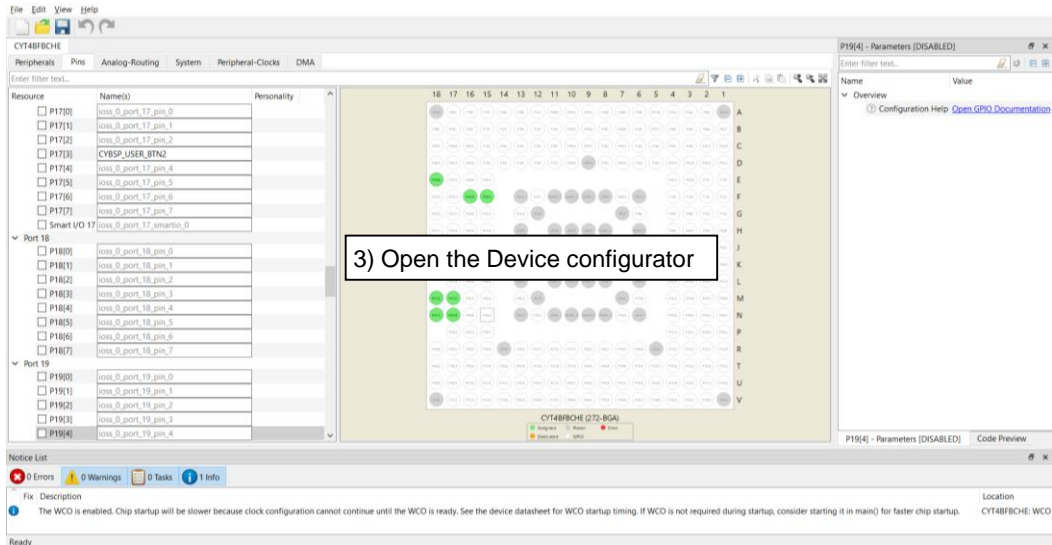
## › Launch the Device Configurator

- 1) Select the “Smart\_IO\_training” project.
- 2) Click the Device Configurator in the Quick Panel
- 3) Then, open the Device configurator window



1) Select “Smart\_IO\_training” project

2) Click the Device configurator

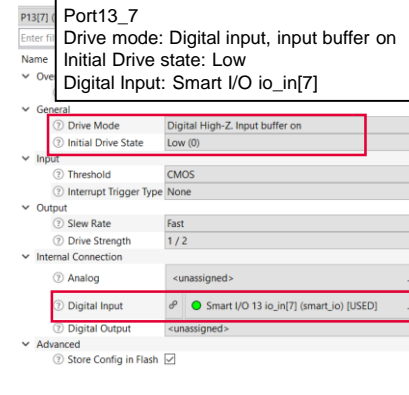
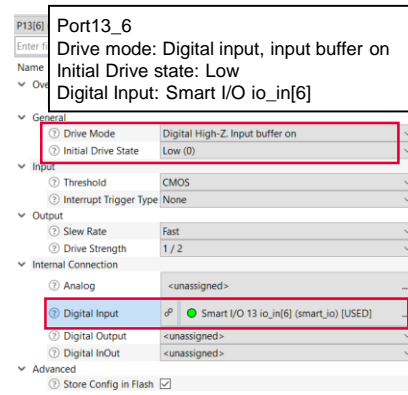
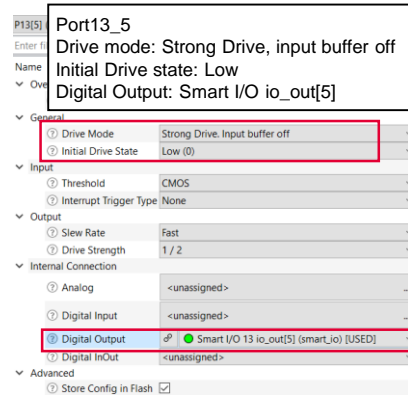
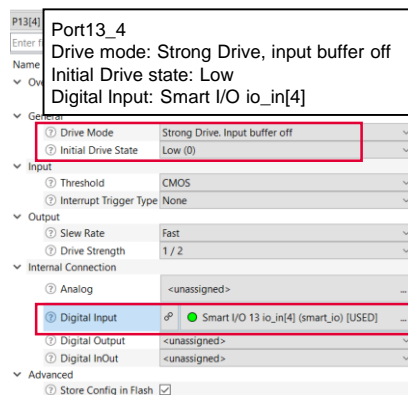
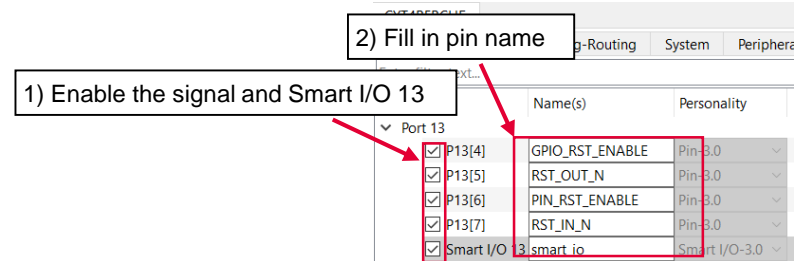


3) Open the Device configurator

# Smart I/O configuration (contd.)

## > Configure GPIO

- In this use case, Smart I/O 13 (Port 13) is used
- Pin assignment is as follows:
  - P13[7] (I/O 7) is used for “RST\_IN\_N”
  - P13[6] (I/O 6) is used for “PIN\_RST\_ENABLE”
  - P13[5] (I/O 5) is used for “RST\_OUT\_N”
  - P13[4] (Chip 4) is used for “GPIO\_RST\_ENABLE”
- Configure GPIO from the pull-down list



# Smart I/O configuration (contd.)

## › Launch the Smart I/O configurator

- 1) Select Smart I/O 13
- 2) Click **Launch Smart I/O Configurator**
- 3) Click **Yes** and then open the Smart I/O Configurator window

The screenshot shows the Infineon Smart I/O configurator software interface. The main window displays a list of resources on the left, including Smart I/O 13, and a grid of pins on the right. A dialog box titled "Device Configurator 4.0" is open in the center, asking "The document has been modified. Do you want to save your changes?" with "Yes" and "Cancel" buttons. A "Launch Smart I/O Configurator" button is highlighted in the top right corner of the main window. A "Smart I/O 13 (smr\_io)" window is also visible at the bottom right.

1) Select Smart I/O 13

2) Click Launch Smart I/O Configurator

3) Click Yes button

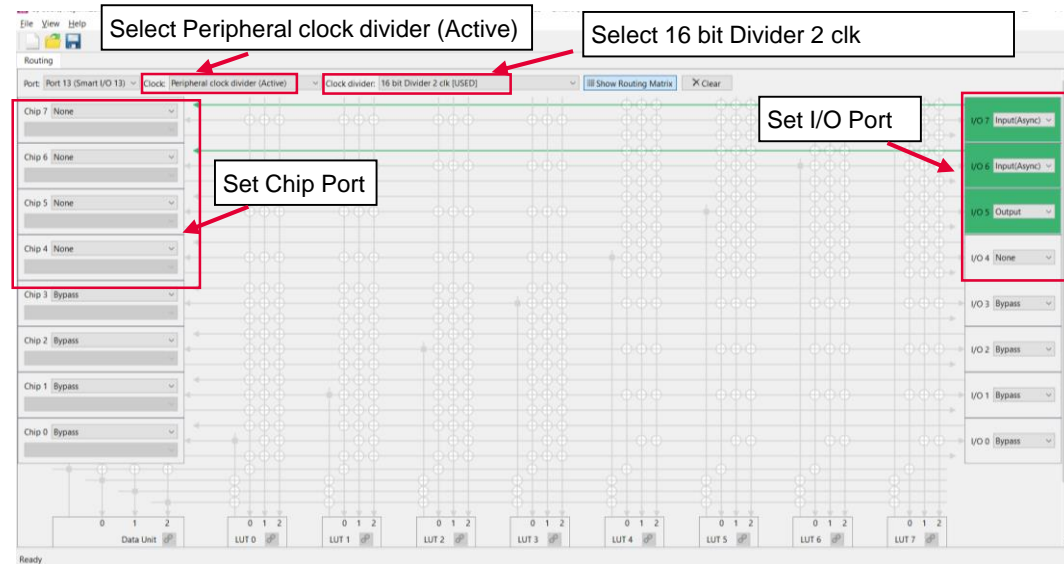
Open the Smart I/O configurator



# Smart I/O configuration (contd.)

## > Configure pin assignment and clock

- Pin assignment is as follows:
  - I/O 7 is used for “RST\_IN\_N”, set to Input (Async)
  - I/O 6 is used for “PIN\_RST\_ENABLE”, set to Input (Async)
  - I/O 5 is used for “RST\_OUT\_N”, set to Output
  - Chip 4 is used for “GPIO\_RST\_ENABLE”, set to none
  - I/O [3:0] is set to bypass.  
These pins do not use the Smart I/O fabric
- Clock setting
  - Set Clock to “Peripheral clock divider (Active)”
  - Set Clock divider to “16 bit Divider 2 clk”



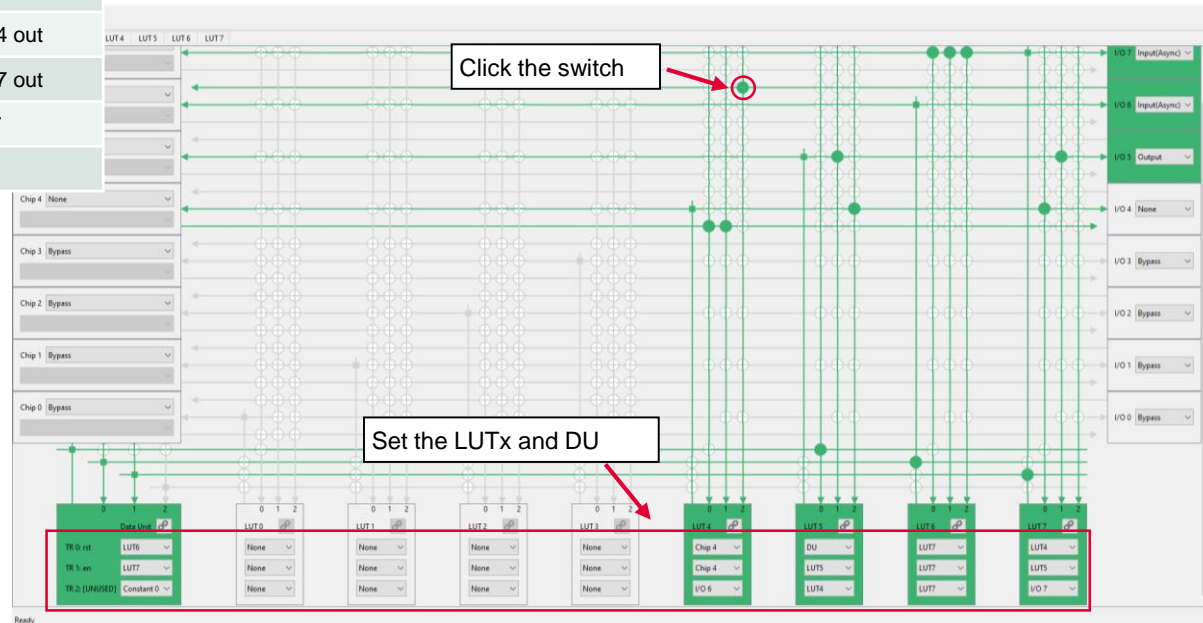
# Smart I/O configuration (contd.)

## > Configure connection of LUTs and DU

- LUTs and DU connect to:

LUTx/DU	TR0	TR1	TR2
LUT4	Chip 4	Chip 4	I/O 6
LUT5	DU out	LUT5 out	LUT4 out
LUT6	LUT7 out	LUT7 out	LUT7 out
LUT7	LUT4 out	LUT5 out	I/O 7
DU	LUT6 out	LUT7 out	"0"

- You can set LUTx and DU using the pull-down lists in LUT and DU box.
- If the routing matrix is shown, you can also click on the switches in the fabric to make input connections.
- All three inputs to the LUT must be designated.  
If the input signal has only one source (for example, LUT6), the same signal inputs to three inputs of LUT.



# Smart I/O configuration (contd.)

## > Configure DU

- Select the Data Unit tab to configure DU

Parameters	Setting
Opcode	Increment and wrap
DATA0	Constant 0
DATA1	Data Register
Register value	127 (Decimal)
Size	8-bit size/width operand

- When the DU in the “Routing” tab is configured to accept an input other than a Constant 0, the corresponding “Data Unit” tab will appear.

Select “Data Unit” tab

File View Help

Data Unit
LUT 4
LUT 5
LUT 6
LUT 7

Opcode: Increment and wrap

DATA0: Constant 0

DATA1: DATA Register

Register Value:  Decimal

Size: 8-bits size/width operand

Details:

Clock = 16 bit Divider 2 clk (CLK\_SmartIO) [USED]  
 TR0 = LUT6  
 TR1 = LUT7

```

du_size = Size - 1
mask = (1 << (du_size + 1)) - 1
data_eq_l1 = (data & mask) == (DATA1 & mask)

Combinatorial:
tr_out = data_eq_data1

Registered/Clocked:
data <= data
if (TR0)
{
  data <= DATA0 & mask
}
else if (TR1)
{
  data <= data_eq_data1 ? DATA0 & mask : (data + 1) & mask
}
          
```

Increment Wrap

TR0 — rst    out — TR\_out

TR1 — en

Clock —> clk

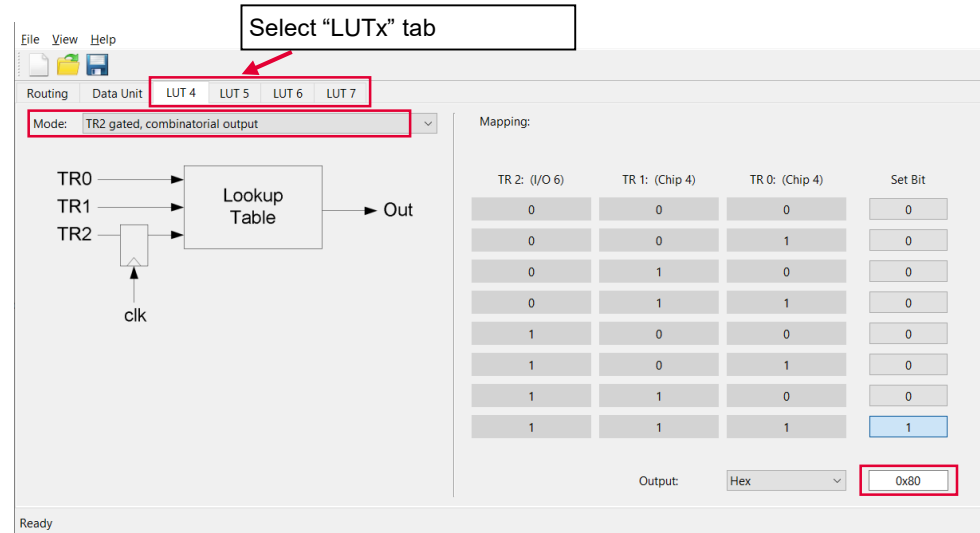
# Smart I/O configuration (contd.)

## > Configure LUT

- Select the Data Unit tab to configure LUTx

LUT	Parameters	Setting
LUT4	Mode	TR2 gated, combination output
	Output	0x80
LUT5	Mode	Sequential (gated) output
	Output	0x60
LUT6	Mode	Combination output
	Output	0x7F
LUT7	Mode	TR2 gated, combination output
	Output	0x28

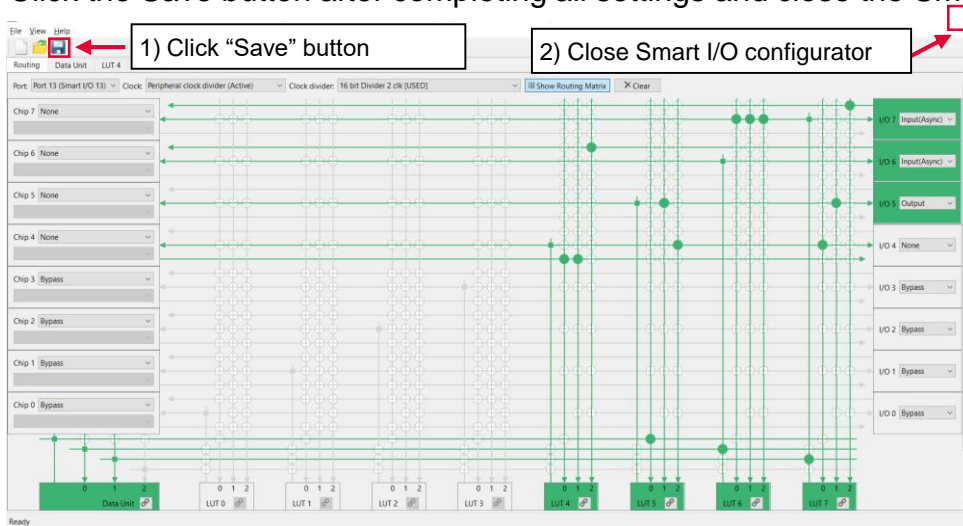
- When an LUT in the Routing tab is configured to accept an input, the corresponding LUT configuration tab will appear.



# Smart I/O configuration (contd.)

## > Close Smart I/O configurator

- Click the Save button after completing all settings and close the Smart I/O configurator



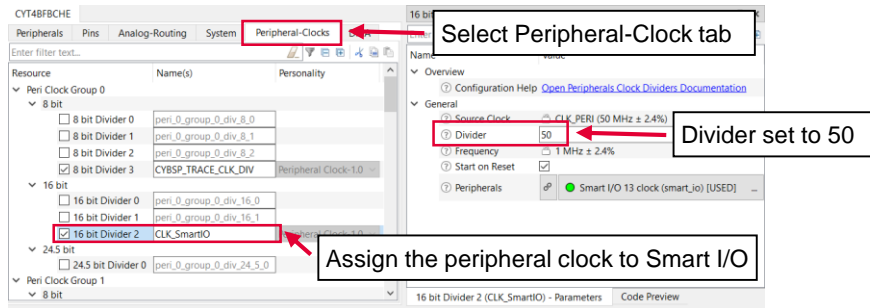
- If an Errors/Tasks message appears, it should be resolved according to the instructions



# Smart I/O configuration (contd.)

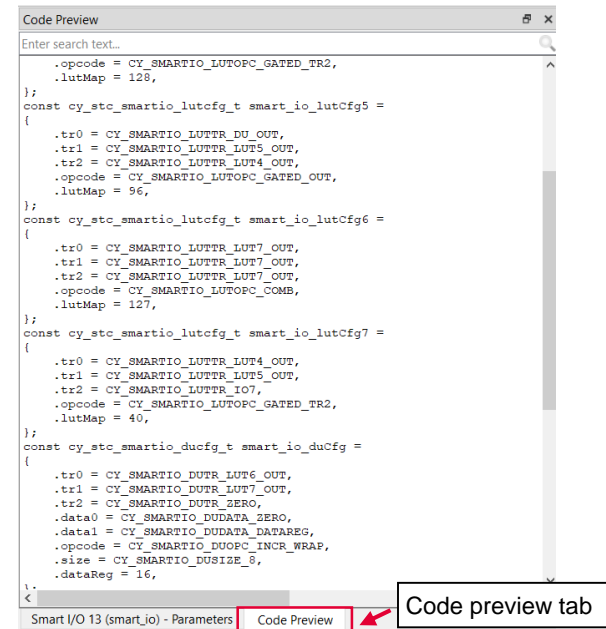
## > Clock configuration

- Peripheral clock configuration in the Device Configurator for Smart I/O resources
- Divides the source clock (50 MHz) by 50



## > Confirm configuration result

- You can check the configuration result in the “Code Preview” tab of the Device Configurator



# Smart I/O configuration (contd.)

## > Close Device configurator

- Click the Save button after completing all settings, then close the Device configurator

The screenshot shows the Smart I/O configurator interface for a CYT48F8CHE device. The interface is divided into several sections:

- Resource List:** A table listing resources such as P12[4-7], Smart I/O 12, Port 13 (P13[0-7]), and Port 14 (P14[0-7]).
- Pin Configuration Matrix:** A grid showing the configuration of pins across ports 13 and 14. Pins are color-coded: green for assigned, orange for reserved, and red for error.
- Parameters Panel:** A panel on the right showing configuration parameters for Smart I/O 13, including Clock Divider (16 bit Divider 2 clk), Hold Override, and I/O Terminal settings.
- Buttons:** A red arrow points to the "Save" button in the top-left corner of the main window. Another red arrow points to the "Close" button in the top-right corner of the "Smart I/O 13 (smart\_io) - Parameters" sub-window.
- Notice List:** A bottom section showing a message: "The WCO is enabled. Chip startup will be slower because clock configuration cannot continue until the WCO is ready. See the device datasheet for WCO startup timing. If WCO is not required during startup, consider starting it in main() for faster chip startup."

# Smart I/O configuration (contd.)

## > Configuration file

- The Smart I/O Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location you saved the \*.modus file for non-IDE applications.
- In this example, the following code is generated:

The screenshot displays the Eclipse IDE's 'GeneratedSource' directory. Two files are highlighted: `cycfg_pins.c` and `cycfg_pins.h`. Red arrows point from these files to their respective code snippets.

**cycfg\_pins.c** code snippet:

```

142 }
143 const cy_stc_smartio_lutcfg_t smart_io_lutCfg6 =
144 {
145     .tr0 = CY_SMARTIO_LUTTR_LUT7_OUT,
146     .tr1 = CY_SMARTIO_LUTTR_LUT7_OUT,
147     .tr2 = CY_SMARTIO_LUTTR_LUT7_OUT,
148     .opcode = CY_SMARTIO_LUTOPC_COMB,
149     .lutMap = 127,
150 };
151 const cy_stc_smartio_lutcfg_t smart_io_lutCfg7 =
152 {
153     .tr0 = CY_SMARTIO_LUTTR_LUT4_OUT,
154     .tr1 = CY_SMARTIO_LUTTR_LUT7_OUT,
155     .tr2 = CY_SMARTIO_LUTTR_LUT7_OUT,
156     .opcode = CY_SMARTIO_LUTOPC_GATED_TR2,
157     .lutMap = 40,
158 };
159 const cy_stc_smartio_ducfg_t smart_io_ducfg =
160 {
161     .tr0 = CY_SMARTIO_DUTR_LUT6_OUT,
162     .tr1 = CY_SMARTIO_DUTR_LUT7_OUT,
163     .tr2 = CY_SMARTIO_DUTR_EREO,
164     .data0 = CY_SMARTIO_DUDATA_EREO,
165     .data1 = CY_SMARTIO_DUDATA_DATAREG,
166     .opcode = CY_SMARTIO_DUOPC_INCR_WRAP,
167     .size = CY_SMARTIO_DUSIZE_8,
168     .dataReg = 16,
169 };
170 const cy_stc_smartio_config_t smart_io_config =
171 {
172     .clkSrc = CY_SMARTIO_CLK_DIVACT,
173     .bypassMask = CY_SMARTIO_CHANNEL0|CY_SMARTIO_CHANNEL1|CY_SMARTIO_CHANNEL2|CY_SMARTIO_CHANNEL3|0x10|0x10,
174     .ioSyncEn = 0x10|0x10|0x10|0x10|0x10,
175     .chipSyncEn = 0x10|0x10|0x10|0x10|0x10,
176     .lutCfg0 = NULL,
177     .lutCfg1 = NULL,
178     .lutCfg2 = NULL,
179     .lutCfg3 = NULL,
180     .lutCfg4 = smart_io_lutCfg4,
181     .lutCfg5 = smart_io_lutCfg5,
182     .lutCfg6 = smart_io_lutCfg6,
183     .lutCfg7 = smart_io_lutCfg7,
184     .ducfg = smart_io_ducfg,
185     .hidOvr = false,
186 };
    
```

**cycfg\_pins.h** code snippet:

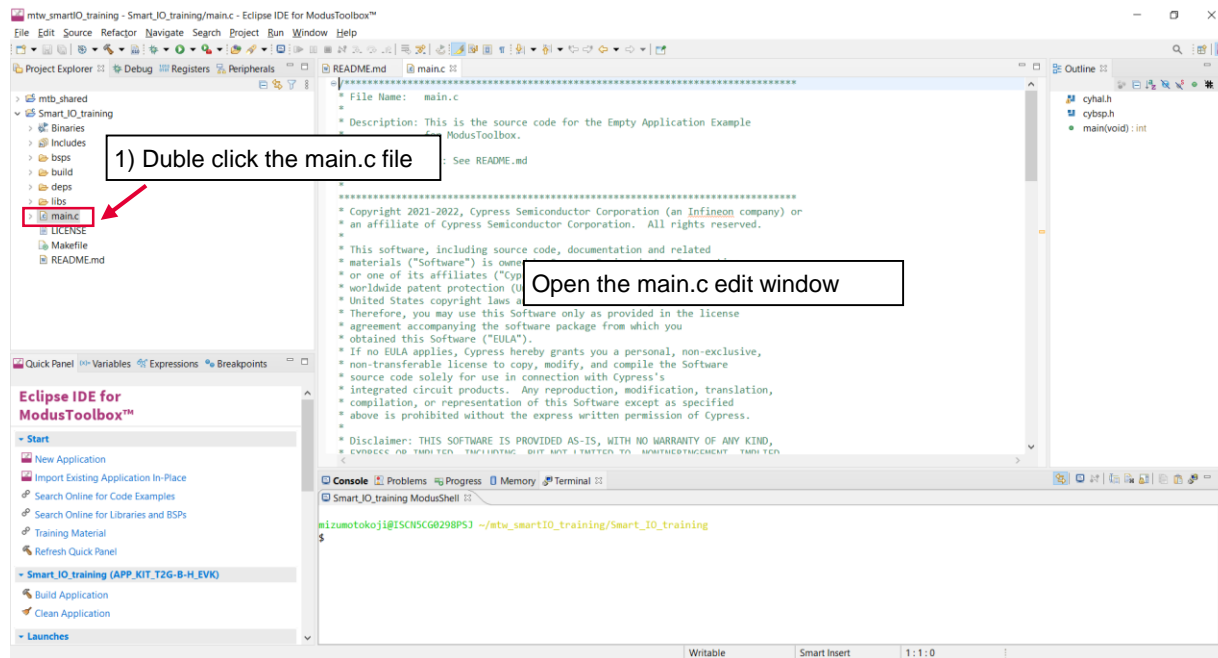
```

73     ... #define CYBSP_D3_CYBSP_DEBUG_UART_CTS
74     #endif //defined (CY_USING_HAL)
75     #define GPIO_RST_ENABLE_ENABLED_IU
76     #define GPIO_RST_ENABLE_PORT GPIO_PRT13
77     #define GPIO_RST_ENABLE_PORT_NUM_I3U
78     #define GPIO_RST_ENABLE_PIN_4U
79     #define GPIO_RST_ENABLE_NUM_4U
    
```



# Implementation

- › This section describes how to implement the configured Smart I/O. This example will implement Smart I/O configuration in the Smart\_IO\_training project.
- Open main.c in Smart\_IO\_training project



# Implementation (contd.)

## › Add include file

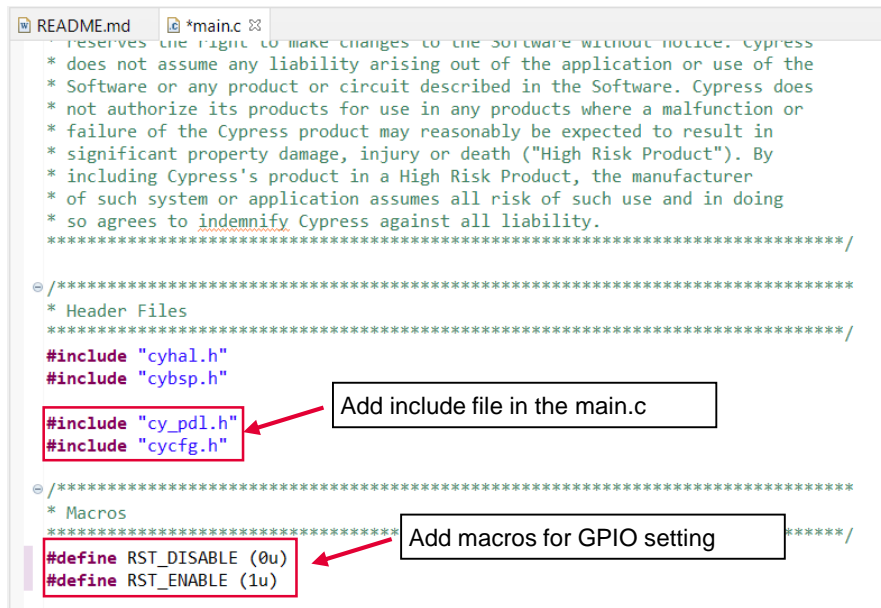
```

README.md  *main.c
reserves the right to make changes to the software without notice. Cypress
* does not assume any liability arising out of the application or use of the
* Software or any product or circuit described in the Software. Cypress does
* not authorize its products for use in any products where a malfunction or
* failure of the Cypress product may reasonably be expected to result in
* significant property damage, injury or death ("High Risk Product"). By
* including Cypress's product in a High Risk Product, the manufacturer
* of such system or application assumes all risk of such use and in doing
* so agrees to indemnify Cypress against all liability.
*****/

/*****
* Header Files
*****/
#include "cyhal.h"
#include "cybsp.h"
#include "cy_pdl.h"
#include "cycfg.h"

/*****
* Macros
*****/
#define RST_DISABLE (0u)
#define RST_ENABLE (1u)

```





## Implementation (contd.)

---

### Smart I/O initialization

- › Call the [Cy\\_SmartIO\\_Init\(\)](#) function to configure Smart I/O
  - Initialize the Smart I/O 13 include LUT4-7 and DU setting

### Smart I/O enable

- › Call the [Cy\\_SmartIO\\_Enable\(\)](#) function to enable Smart I/O

### GPIO port write

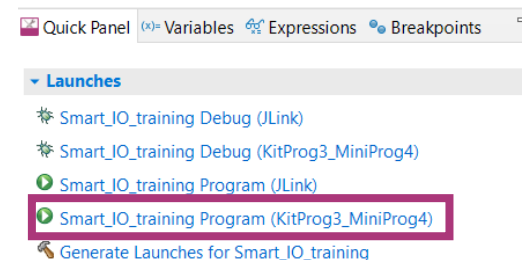
- › Call the [Cy\\_GPIO\\_Write\(\)](#) function to set GPIO
  - It is used to enable the “GPIO\_RST\_ENABLE” signal
  - “GPIO\_RST\_ENABLE” is configured as “**GPIO\_RST\_ENABLE\_PORT** (= Port 13)” and “**GPIO\_RST\_ENABLE\_PIN** (= 4 pin)” in ***cycfg\_pins.h*** file

# Compiling and programming

1. Connect to power and USB cable
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming
3. Compile
  - a) Select the target application project in the Project Explorer
  - b) In the Quick Panel, scroll down, and click “Build Application” in Smart\_IO\_training (APP KIT\_T2G-B-H\_EVK)

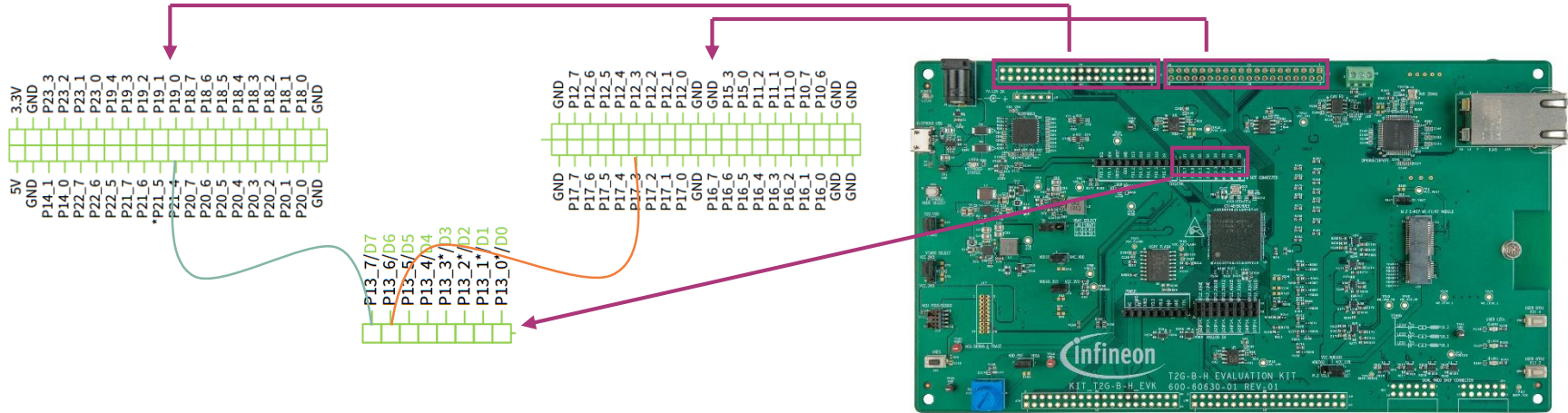


4. Programming
  - a) Select the target application project in the Project Explorer
  - b) In the Quick Panel, scroll down, and click “Smart\_IO\_training Program (KitProg3\_MiniProg4)” in Launches



# Run and test

- To run this use case, use jumper wires, as shown below, on the board



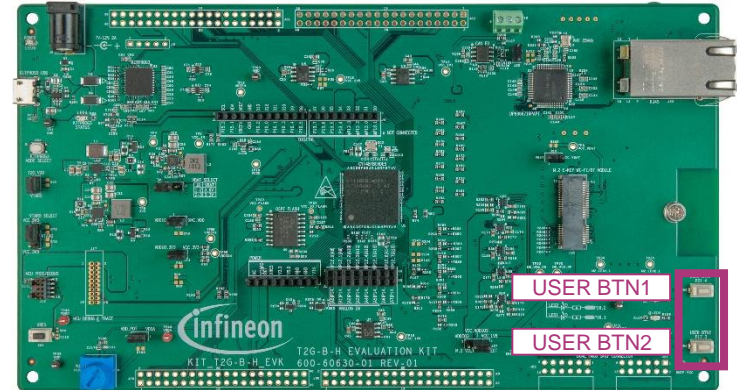
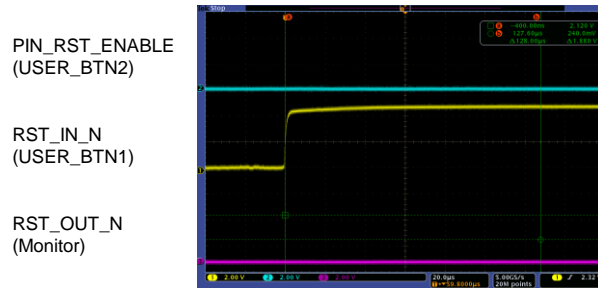
Name	GPIO pin	Connection	Function
PIN_RST_ENABLE	Port13.6	Port 17.3	USER_BTN2
RST_IN_N	Port13.7	Port 21.4	USER_BTN1
RST_OUT_N	Port13.5		Monitor

## Run and test (contd.)

2. After programming, the application starts automatically
3. Press USER BTN1 (RST\_IN\_N), you can observe the following waveform



4. When press USER BTN1 (RST\_IN\_N) while pressing USER BTN2 (PIN\_RST\_ENABLE), the output does not change.



# References

---

## Datasheet

- › [CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family](#)

## Architecture Technical reference manual

- › [TRAVEO™ T2G automotive body controller high family architecture technical reference manual](#)

## Registers Technical reference manual

- › [TRAVEO™ T2G Automotive body controller high registers technical reference manual](#)

## PDL/HAL

- › [PDL](#)

- › [HAL](#)

## Training

- › [TRAVEO™ T2G Training](#)

## Application note

- › [Smart I/O Usage Setup in Traveo II Family](#)



# Revision History

---

Revision	ECN	Submission Date	Description of Change
**	7844899	2022-12-07	Initial release

# Important notice and warnings

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2022-12**

**Published by  
Infineon Technologies AG  
81726 Munich, Germany**

**© 2022 Infineon Technologies  
AG.**

**All Rights Reserved.**

**Do you have a question about  
this document?**

**Go to:**  
[www.infineon.com/support](http://www.infineon.com/support)

**Document reference  
002-36605 Rev. \*\***

## **IMPORTANT NOTICE**

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenhheitsgarantie”).

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office ([www.infineon.com](http://www.infineon.com)).

## **WARNINGS**

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.